VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019 Šimon Orság

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Šimon Orság**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: IT Lab czech s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Marek Běhálek, Ph.D.**

Konzultant bakalářské práce: David Maralík

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2.4.2019 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 2.4.2019 .....

Rád bych na tomto místě poděkoval firmě IT Lab czech s.r.o. za nabídku odborné praxe a také za odborný dohled při vytváření mé bakalářské práce. Dále bych rád poděkoval panu Ing. Markovi Běhálkovi, Ph.D. za konzultace a pomoc v průběhu odborné praxe.

Abstrakt

Bakalářská práce popisuje průběh individuální odborné praxe ve firmě IT Lab czech s.r.o., ve které jsem pracoval na pozici back-end a front-end developer. Práce je rozdělena do několika částí, první část obsahuje představení firmy a pracovní zařazení studenta. V druhé části jsou popsány zadané úkoly a v poslední části je celkové shrnutí průběhu praxe a také přínosy této praxe.

Klíčová slova: JavaScript, React, Redux, Yarn, HTML5, CSS3, SASS, PHP, Symfony, Sylius, Git

Abstract

This bachelor thesis describes the course of individual professional practice in the company IT Lab czech s.r.o, in which I worked as Back-End and Front-End developer. The work is divided into several parts, the first part contains the company presentation and student's job classification. The second part describes individual tasks and in the last part is overall summary of the practise and also the benefits of this practise.

Key Words: JavaScript, React, Redux, Yarn, HTML5, CSS3, SASS, PHP, Symfony, Sylius, Git

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
APP	–	Application
CSS	–	Cascading Style Sheets
DOM	–	Domain Object Model
HTML	–	Hyper Text Markup Language
JS	–	JavaScript
SASS	–	Syntactically Awesome Style Sheets
PHP	–	Hypertext Preprocessor
REST	–	Representational State Transfer
UX	–	User Experience

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Moderní technologie tvoří nedílnou součást webových stránek a aplikací. Jsou základem pro robustní a funkční řešení. V dnešní době se můžeme s těmito technologiemi setkat prakticky všude. Využitím moderních technologií se také zabývá firma IT Lab czech s.r.o, ve které jsem měl možnost se s těmito technologiemi seznámit a vyzkoušet si jejich reálné využití.

V první části (kapitola 2) bakalářské práce se budu věnovat představením firmy, stručného přehledu činnosti firmy a také podrobněji popíšu svou pracovní pozici. V další kapitole (kapitola 3) popíšu jednotlivé technologie, které byly použity při vypracování zadaných úkolů.

Druhou část (kapitola 4-6) věnuji přehledu zadaných úkolů a také jejich samotné tvorby. Větší část je zaměřena na hlavní úkol praxe, kterým bylo seznámení s Reactem a implementace komponent webové aplikace FoxDeli. Zde popíšu detailněji tvorbu komponent v JavaScriptovém frameworku React a také řešení problémů, se kterými jsem se v průběhu setkal. Menší část je zaměřena na tvorbu modulu e-shopového frameworku Sylius. Zde popíšu průběh tvorby modulu a vytvoření balíčku.

V poslední části (kapitola 7) zhodnotím průběh praxe, teoretické a praktické znalosti získané v průběhu studia, a také dovednosti, které mi při praxi scházely.

2 Představení společnosti a pracovní zařazení

2.1 Základní popis společnosti

Společnost IT Lab vznikla v roce 2011 v Technologickém Inovačním centru ČKD v Praze se zaměřením na vývoj webových stránek a internetových obchodů. V roce 2013 otevřela pobočku ve Vědecko-Technologickém parku Ostrava. V současné době společnost sídlí v centru Ostravy.

Během své existence změnila zaměření a dnes se především orientuje na vývoj softwarových produktů pro startupy a inovativní společnosti v rané fázi.

IT Lab se aktuálně zabývá hlavně kompletním outsourcing vývojem softwarových produktů, vývojem multi-channel platforem pro e-commerce, vývojem internetových obchodů, systémovými integracemi a také tvorbou webových portálů.

Hlavními technologiemi, které firma při vývoji používá jsou PHP - Nette / Symfony, JS - Node.js a React, Ruby on Rails, Swift (iOS), MySQL, MongoDB, Redis, RabbitMQ, Elastic-Search a další.

2.2 Pracovní zařazení studenta

Ve firmě IT Lab jsem pracoval na pozici front-end developera, později jsem měl možnost si také vyzkoušet pozici back-end developera.

Ve "front-end" části jsem pracoval s technologiemi, jako jsou JavaScript (React), CSS (SASS) a také HTML5. V druhé části praxe jako back-end developer jsem měl možnost se seznámit s e-shop platformou Sylius, pro kterou jsem vytvořil synchronizační modul pro balíkovou službu FoxDeli. Během praxe jsem pracoval spolu s dalšími lidmi z týmu, se kterými jsem konzultoval všechny návrhy a problémy.

3 Použité technologie

3.1 JavaScript

JavaScript je dynamický, objektově orientovaný skriptovací jazyk, který se používá primárně pro webové stránky, ale může být také součástí serverových aplikací, např. MongoDB, CouchDB a Node.js. [1]

JavaScript je multiplatformní dynamický jazyk podporující objektově orientované programování. Při použití JavaScriptu na webové stránce probíhá interpretace přímo ve webovém prohlížeči u klienta. [2]

3.2 HTML

HTML je značkovací jazyk, který slouží pro tvorbu webových stránek. HTML používá definované značky (tzv. tagy) a jejich vlastnosti (atributy) k vytváření jednotlivých elementů (např. *<header>, <body>, <footer>, <title>*). Každá značka musí mít otevírací a ukončovací znak (používají se úhlové závorky < a >). [3, 4]

3.3 CSS

CSS je stylovací jazyk, který se používá pro vytvoření stylu webové stránky. Je standardizován dle specifikace W3C. [5, 6]

3.4 SASS

SASS je skriptovací jazyk, který slouží jako rozšíření funkcionality CSS. SASS obsahuje navíc několik důležitých součástí - proměnné, funkce, cykly, argumenty a dědění vlastností.

SASS se pomocí kompilátoru musí zkompilovat na CSS soubor, kterému rozumí prohlížeč. [7]

3.5 Bootstrap

Bootstrap je balíček nástrojů pro tvorbu webu a webových aplikací. Obsahuje kompletně předpřipravené HTML elementy a CSS styly. Balíček také obsahuje responzivní design, který umožňuje měnit rozložení stránky dynamicky na základě používaného zařízení (PC, tablet, mobilní telefon). Pro použití Bootstrapu by měl mít uživatel alespoň základní znalost HMTL a CSS. [8]

3.6 ESLint

ESLint je open-source JavaScript analyzátor kódu, který pomocí předdefinovaných pravidel zkontroluje kód a poté vyhodnotí případné chyby.

U více členných týmů se běžně stává, že každý používá svou strukturu a pravidla pro psaní kódu, z toho důvodu se používá ESLint, který zamezí případným problémům. [9]

3.7 Yarn

Yarn je správce balíčků pro JavaScriptové aplikace. Umožňuje sdílení kódu s ostatními vývojáři pomocí balíčků (modulů), které lze do aplikace jednoduše přidat. Také poskytuje aktualizace na případně novější verze balíčků.

Yarn používá soubor package.json, ve kterém jsou veškeré informace o balíčcích, které aplikace aktuálně používá pro produkční a vývojové prostředí (dev). [10]

3.8 React

React je JavaScriptová open-source knihovna pro vytváření webových komponent. Umožňuje nám aplikaci rozdělit do několika znovupoužitelných komponent, které mají svůj stav, data a metody. React slouží jako základ pro jednostránkové nebo mobilní aplikace.

Autorem knihovny je společnost Facebook, která dříve tuto knihovnu používala interně pro své účely.

React aplikace většinou používají dodatečné knihovny pro routování, správu stavů (dat) a komunikaci s externími API. Poté se jedná o tzv. komplexní aplikaci, která obsahuje veškerou logiku, HTML, CSS v jednom JavaScriptovém balíčku. [11, 12, 13]

3.9 Redux

Redux je JavaScriptová open-source knihovna pro správu stavů aplikace. Jedná se o centrální úložiště (stavový kontejner) aplikace, které je pouze dočasné a po znovu načtení aplikace je vše smazáno.

Redux umožňuje ukládat globální stavy komponent/aplikace a také umožňuje případné změny stavů oznamovat všem naslouchajícím komponentám.

Většinou se používá v kombinaci s React nebo Angular knihovnou. [14, 15]

3.10 Apollo Client

Apollo je JavaScriptová knihovna, která umožňuje komunikovat s GraphQL serverem (API). Také umožňuje získaná data dočasně uchovávat (cachovat). [16]

3.11 GraphQL

GraphQL je dotazovací jazyk pro API rozhraní. Funguje na úrovni aplikační vrstvy. Umožňuje získat pouze žádaná data, tím se odlišuje od klasického REST API.

Podporuje také vlastní datové struktury / typy, vnořenou strukturu, volání funkcí (např. přihlášení / odhlášení uživatele), zabezpečení určitých částí dat. Také je možné kombinovat více dotazů do jednoho komplexnějšího. Typový systém také umožňuje validaci požadavku před samotným vykonáním, to zajistí správnost dat a také snižuje zátěž serveru. [17, 18]

3.12 Babel

Babel je kompilátor JavaScript kódu, který převádí novou syntaxi JavaScriptu na starý ES5 standart, který umožňuje běh aplikace i ve starších prohlížečích. Jedná se hlavně o specifikaci ES6 a výše, ve které je možné používat třídy, iterátory, psát funkce zkráceně (arrow funkce) a mnoho dalšího. [19]

3.13 Webpack

Webpack je module JavaScript bundler. Jednoduše řečeno, jedná se o proces spojení všech modulů a souborů do jednoho, poté je možné spustit výslednou aplikaci v prohlížeči. [20, 21]

3.14 GIT

GIT je verzovací systém, který umožňuje správu verzí souborů a také jednoduchou integraci týmu lidí při spolupráci na stejném projektu / kódu atp. [22, 23]

3.15 Symfony

Symfony je open-source PHP framework pro vytváření webových stránek a aplikací. Je založený na principu návrhového vzoru MVC (model-view-controller). Jedná se o balík komponent, které nám zjednoduší vývoj samotné aplikace. Také umožňuje jednoduché rozšíření o další knihovny komponent. [24, 25]

3.16 Sylius

Sylius je e-commerce open-source PHP framework, který je postavený na Symfony frameworku. Jedná se o balík komponent a funkcí, které zjednoduší vývoj e-shopu.

Součástí je administrační rozhraní, šablonovací systém, správa objednávek a dopravy, produktový katalog, podpora multiměn, lokalizace, platební metody, správa zákazníků apod. [26]

4 Seznam zadaných úkolů

4.1 FoxDeli komponenty

Úkolem bylo vytvořit stránku s přehledem zásilek, včetně stránkování, filtrů a hromadných akcí v již existující aplikaci FoxDeli.

Jednotlivé části úkolu a jejich časová náročnost:

- Analýza rozdělení komponent [4 dny]
- Seznámení s Reactem [5 dní]
- Návrh jednotlivých komponent přehledu zásilek [7 dní]
- Návrh stránkovací komponenty [4 dny]
- Stylování komponent [2 dny]
- Složení komponent do funkčního celku [6 dnů]
- Testování [4 dny]

Cílem je správně navrhnout a rozdělit jednotlivé komponenty, které se budou vyskytovat na stránce, následně je graficky upravit dle návrhu a otestovat celkovou funkčnost stránky.

4.2 Sylius modul

Úkolem bylo vytvořit modul do e-commerce frameworku Sylius, který by umožnil synchronizaci zásilek přes FoxDeli API.

Jednotlivé části úkolu a jejich časová náročnost:

- Analýza postupu [4 dny]
- Seznámení se Sylius frameworkem [5 dní]
- Seznámení s API FoxDeli [4 dny]
- Návrh struktury modulu [3 dny]
- Implementace modulu [3 dny]
- Testování [2 dny]

Cílem je propojit Sylius s FoxDeli API a tím zjednodušit proces vkládání zásilek a generování štítků pro vybrané zásilky bez nutnosti ručního vkládání do FoxDeli.

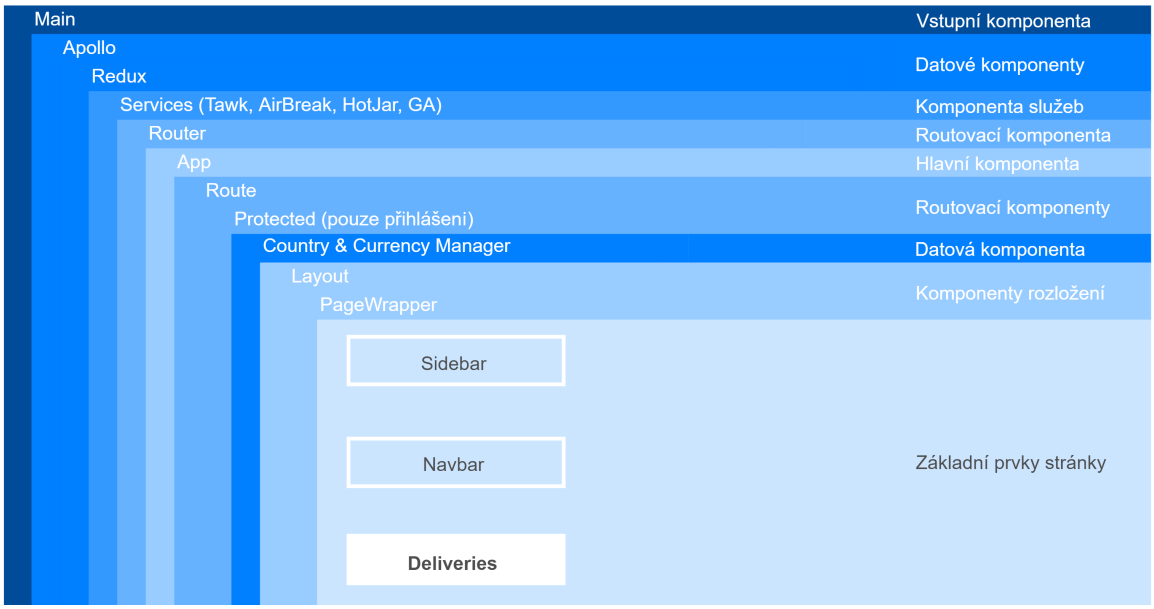
5 FoxDeli komponenty

FoxDeli je webová aplikace pro snadnou správu zásilek e-shopu. Umožňuje importovat nebo ručně vkládat zásilky přes jednoduché webové rozhraní a API. Nastavit personalizované e-maily, SMS a upozornění. Spravovat kontakty nebo si nakonfigurovat stránku pro sledování zásilky.

Aplikace běží na JavaScript frameworku React. Výhodou tohoto řešení je rozdělení jednotlivých částí aplikace do tzv. komponent, které lze jednoduše znovu použít v jiné části aplikace, bez nutnosti psát podobnou funkcionalitu znovu. Pod slovem komponenta si můžeme představit jednoduchou funkci / třídu, která obsahuje logiku pro zpracování a zobrazení dat. Syntaxe pro použití komponenty je podobná HTML - *<MyComponent ...>...</MyComponent>*.

React poté tyto komponenty vloží do svého virtuálního DOMu, který se při jakékoliv změně porovnává se skutečným DOMem a když najde rozdíly, tak je co nejefektivnějším způsobem aktualizuje - vždy pouze části, které se změnily. My se staráme pouze o data, které předáváme jednotlivým komponentám.

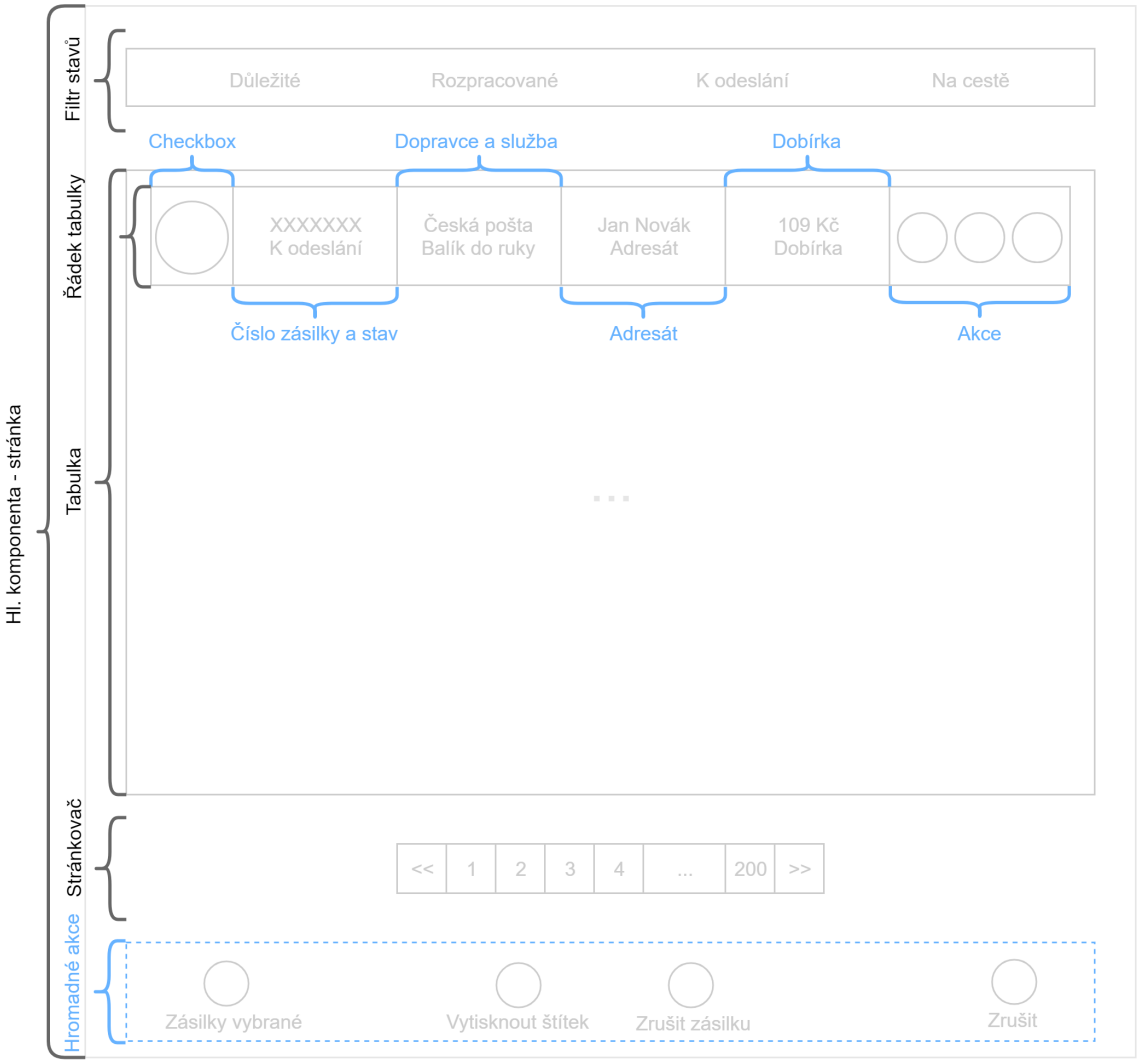
Komponenty se dělí na dva typy, a to stavové a bezstavové. Stavová komponenta uchovává uvnitř sebe svůj stav (state), který lze použít v rámci komponenty, případně je možné tento stav předat dalším komponentám. Bezstavová komponenta pouze stav přebírá přes vstupní parametry (properties, zkráceně props), neumí nic jiného než data zpracovat a zobrazit. Každá komponenta musí obsahovat render metodu, která slouží pro zobrazení výsledných dat. V případě bezstavové komponenty nahrazuje render metodu funkce, která data vrací.



Obrázek 1: Struktura aplikace - vnoření jednotlivých komponent

5.1 Analýza rozdělení komponent

Před tvorbou komponent bylo nutné správně rozdělit stránku se seznamem zásilek na React komponenty. Na základě UX návrhu jsem navrhl, z jakých komponent by se měla stránka skládat. Níže na obrázku je zobrazena struktura. Celkově jsem rozdělil stránku do 12 samostatných komponent, z toho 7 je bezstavových (označené modře).



Obrázek 2: Návrh rozdělení komponent (modře jsou označeny bezstavové)

5.2 Zdroj dat

Celá aplikace používá jako zdroj dat GraphQL API. Pomocí této technologie API jsme schopni získat pouze námi potřebná data v aplikaci (tzn. můžeme si vybrat jaká data nám API vrátí). GraphQL API generuje také dokumentaci, ve které je možné si zobrazit dostupné dotazy, funkce a datové typy včetně jejich dostupných vlastností. O datovou část se staral kolega, tudíž zdroj dat jsem měl předpřipravený. Níže na obrázku je ukázka dotazu a dokumentace GraphQL API.

Obrázek 3: GraphQL API dotaz a dokumentace

Po získání dat je nutné data naformátovat pro použití v dalších komponentách. K tomu nám slouží funkce *formatQueryData*, která data naformátuje do námi dané struktury. Zavolání funkce pro formátování dat probíhá v *componentDidMount()* tzn. okamžitě poté, co byla komponenta zavolána k použití, ale také při změně stránky nebo filtru. Poté se naformátovaná data uloží do stavu (state) hl. komponenty, aby je bylo možné předávat do ostatních komponent a zajistit tak propisování změn.

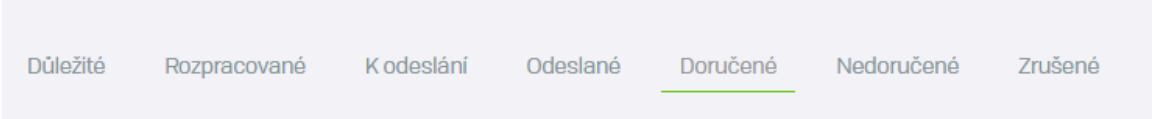
5.3 Tvorba komponent

Hlavní komponenta slouží k "obalení" všech ostatních komponent, které se v ní vyskytují. Jednoduše řečeno, jedná se o stránku, která obsahuje jednotlivé prvky (filtr, tabulku, stránkovač, ...). Aby bylo možné tuto komponentu uživateli zobrazit na určité URL adrese, musí se přidat do konfiguračního souboru routeru (komponenta, která řeší na jaké URL se má zobrazit která komponenta). Tím je zaručeno, že je možné stránku s přehledem zásilek zobrazit na určité URL adrese (/deliveries) jako celek.

```
...
{
    // Pro přístup ke komponentě musí být uživatel přihlášen (protectedRoute())
        , komponenta používá hlavní rozložení aplikace (LayoutMain)
    component: protectedRoute(withLayout(containers.Deliveries, LayoutMain)),
    exact: true, // Vyžadujeme přímou shodu URL adresy
    path: '/deliveries', // URL adresa
}
```

Výpis 1: Ukázka konfigurace routeru

Filtr stavů je jednoduchá komponenta, která při kliknutí na danou položku filtru zavolá funkci, která změní globální stav filtrů, poté dojde k obnově dat o zásilkách. Obnova dat zásilek se řeší ve funkci *componentDidUpdate(prevProps)*, která se automaticky volá při každé změně stavu komponenty. Ve funkci se ověřuje, zda aktuální filtr je odlišný od předchozího stavu filtru, aby se zamezilo zacyklení a také zbytečným požadavkům na GraphQL API.

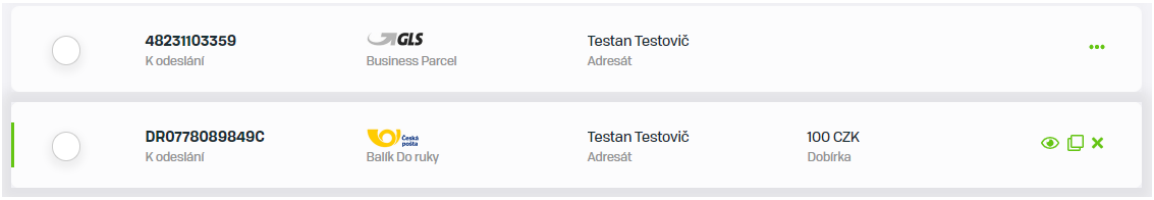


Obrázek 4: Vzhled komponenty filtru

Samotná tabulka se skládá ze dvou komponent, jedna z nich je stavová a druhá bezstavová. Stavová komponenta se stará o uchování stavů, např. zda je tabulka ve stavu načítání dat nebo jaký je aktuálně vybraný řádek. Také předává bezstavové komponentě určité informace - řádky a sloupce, akce, nastavení. Bezstavová komponenta zpracovává řádky a sloupce, které jí předá stavová komponenta a také se stará o předání informací do dalších komponent.

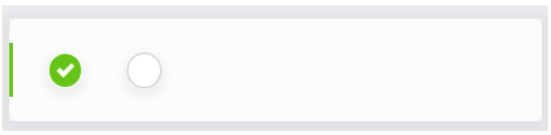
Obrázek 5: Vzhled komponenty tabulky (řádky a sloupce)

Řádek tabulky je bezstavová komponenta, která se stará o vykreslení jednotlivých řádků tabulky. Také umožňuje vkládat vlastní akce (odkaz / funkce) do speciálního sloupečku s akcemi. Důležitou součástí řádku jsou také funkce, které při vykonání určité akce informují nadřazené komponenty. Například při kliknutí na daný řádek je možné vykonat vlastní funkci nebo uživatele přesměrovat na jinou stránku (komponentu). Při najetí na řádek je také volána funkce, která informuje komponentu tabulky o tom, na kterém řádku se uživatel aktuálně nachází. Tato funkcionalita primárně slouží ke zvýraznění vybraného řádku a také k "rozbalení" sloupce s akcemi.



Obrázek 6: Vzhled komponenty řádku - před a po najetí myší

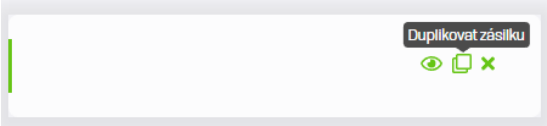
Checkbox je bezstavová komponenta řádku tabulky a slouží k hromadným operacím tabulky. Jako vstupní vlastnosti přebírá pouze zdali je zaškrtnutá, co se má stát při kliknutí a také jestli je možné na ni kliknout (vlastnost disabled).



Obrázek 7: Vzhled komponenty checkbox - zaškrtnutá a nezaškrtnutá

Akce jsou bezstavové komponenty řádku tabulky, které slouží k zobrazení tlačítek pro vykonání akcí pro vybraný řádek. Akce mohou být dvojího typu, a to funkce nebo odkaz. Funkce slouží primárně k zobrazení jiné komponenty (např. tisk štítku), případně k vykonání určité operace na pozadí (např. změna stavu). Odkaz umožňuje uživatele přesměrovat na určitou URL adresu, např. detail zásilky. Jako vstupní vlastnosti přebírají obě komponenty ikonku a tooltip

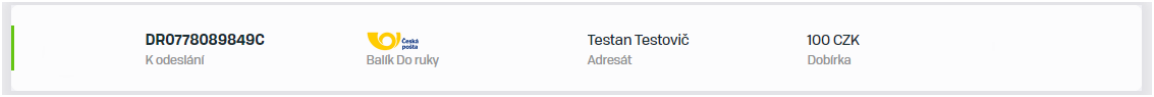
(nápověda při najetí myší na ikonu). Dále vlastnost "to" (URL pro přesměrování) pro typ odkaz a "onClick"(funkce při kliknutí) pro typ funkce.



Obrázek 8: Vzhled komponenty akce

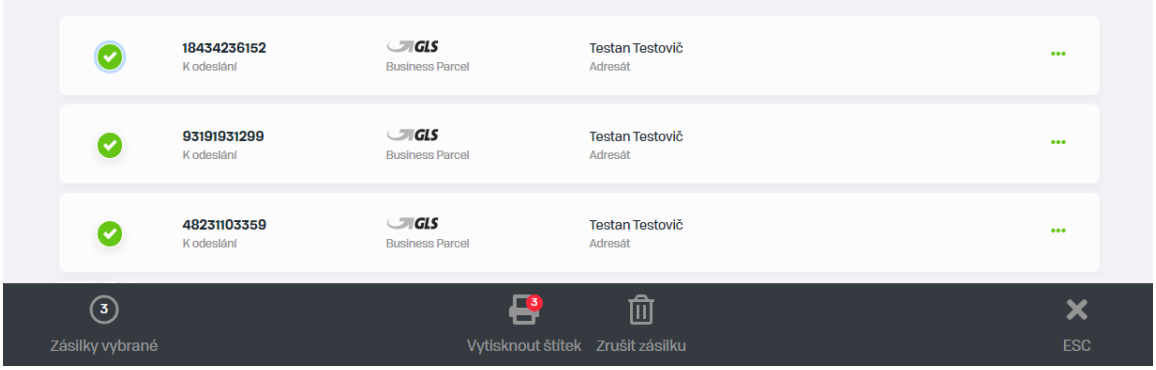
Do ostatních komponent řádku tabulky jsem zařadil všechny, které pouze vypisují formátovaný text. Z toho důvodu není nutné detailně popisovat jak výpis textu probíhá. Vstupní vlastnosti jsou pouze obaleny HTML a poté zobrazeny. Jedná se o komponenty:

- Číslo zásilky a stav - zobrazuje číslo zásilky a stav (např. k odeslání, na cestě)
- Doprava a služba - zobrazuje logo dopravce a typ služby (např. Balík do ruky - Česká pošta, Bussiness parcel - GLS)
- Adresát - jméno a příjmení adresáta (např. Pavel Novák)
- Dobírka - zobrazí se pouze v případě, že je balík na dobírku (hodnota XXX CZK/EUR).



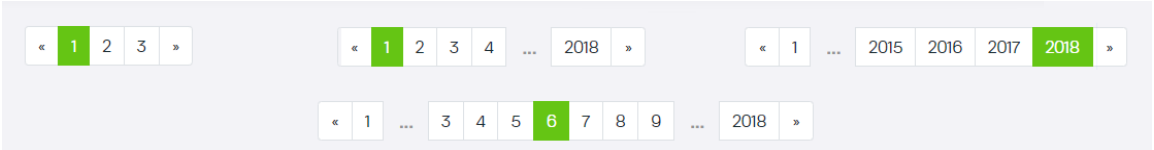
Obrázek 9: Vzhled ostatních komponent - číslo zásilky a stav, doprava a služba, adresát, dobírka

Komponenta hromadné akce je bezstavová komponenta, která umožňuje rychlý a pohodlný způsob, jak pracovat s více zásilkami naráz. Hromadné akce se automaticky zobrazí ve spodní části stránky, v případě že uživatel vybere alespoň jednu zásilku pomocí komponenty checkbox. Komponenta umožňuje hromadný tisk štítků zásilek, případně vybrané zásilky zrušit. Také zobrazuje celkový počet vybraných zásilek. Hromadné operace jsou dostupné pouze pro zásilky stejného dopravce (např. nelze tisknout štítek pro Českou poštu a zároveň GLS) a také pouze pro zásilky, které jsou ve stavu k odeslání (nelze zrušit zásilku, která byla již doručena nebo zrušena). Výběr zásilek lze jednoduše zrušit kliknutím na tlačítko ESC nebo stisknutím klávesy ESC, tím dojde k vyprázdnění seznamu vybraných zásilek a schování hromadných akcí.



Obrázek 10: Vzhled komponenty hromadné akce - vybrání více zásilek

Stránkovač je stavová komponenta, která slouží ke stránkování dat a omezení zobrazených položek na stránce. Vstupními parametry jsou funkce pro oznámení změny stránky, celkový počet stránek a počet "sousedů"- maximální počet stránek, které se zobrazí vlevo a vpravo od aktuálně vybrané stránky, ve výchozím stavu se zobrazují 3. Komponenta poté dle počtu stránek zobrazí rozložení tlačítek pro výběr stránek. Při kliknutí na číslo stránky dojde k zavolání funkce pro oznámení změny stránky, a tím dojde v nadřazené komponentně k získání nových dat nebo filtraci stávajících dat. Pokud je počet stránek menší jak 2, stránkovač se automaticky schová. Stránkovač si uchovává ve svém stavu informace o aktuální stránce, celkovém počtu stránek a počtu "sousedů".



Obrázek 11: Vzhled komponenty stránkovače - několik variant dle počtu stránek

5.4 Stylování komponent a vzhled

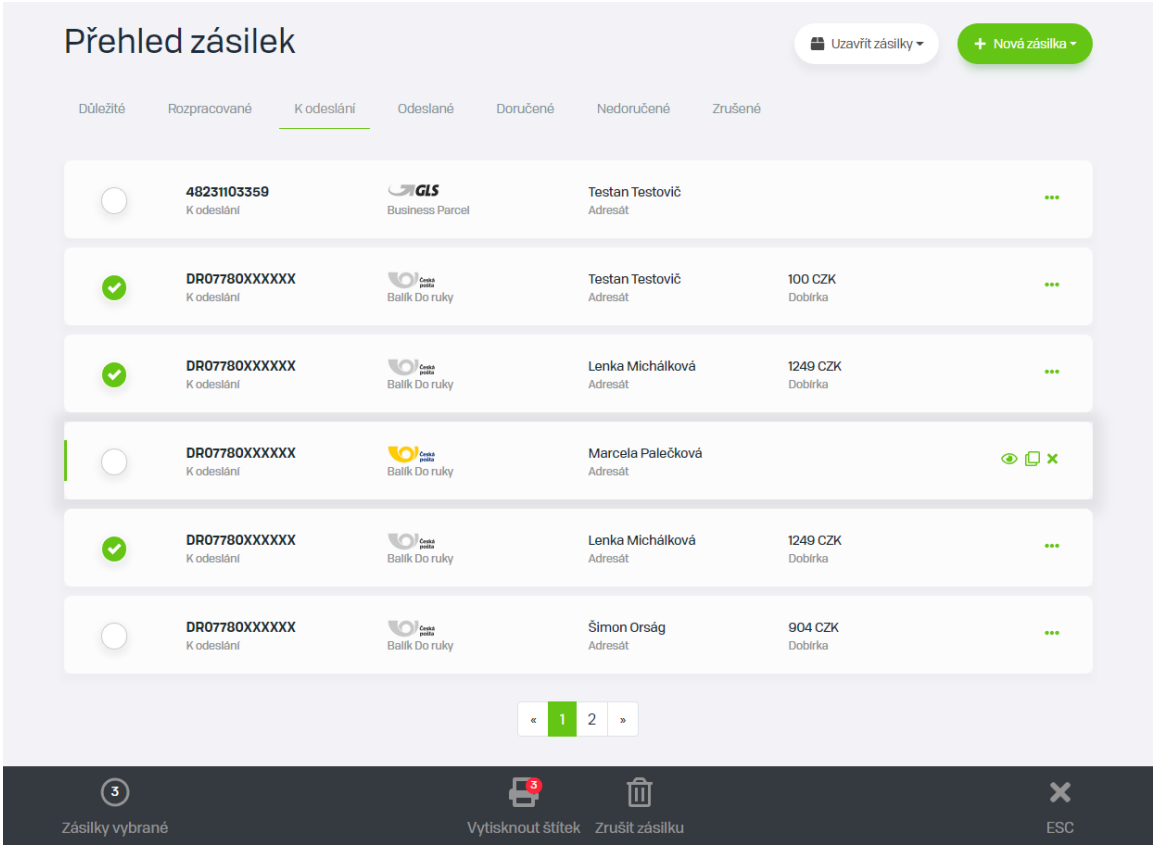
Po dokončení jednotlivých komponent bylo potřeba vše nastylovat. Pro CSS stylování se používal preprocesor SASS. Veškeré návrhy byly předpřipravené UX týmem, který se staral o rozložení a celkový vzhled aplikace. Uživatelské rozhraní bylo vytvořeno tak, aby bylo minimalistické a zároveň příjemné pro uživatele.

Aplikace má být reponzivní, z toho důvodu se používá Bootstrap knihovna, která již obsahuje důležité prvky pro vytvoření responzivní aplikace. Nejdříve se vzhled aplikace řešil pro obrazovky počítače, poté se přidávala jednotlivá pravidla pro mobilní zařízení.

5.5 Shrnutí

Hlavním cílem bylo, aby stránka s přehledem zásilek splňovala všechny funkční požadavky. Tyto požadavky jsem z mého hlediska splnil, jednotlivé prvky jsem rozdělil do komponent, které se dají znovu použít a splňují všechny požadavky. Dalším cílem bylo komponenty nastylovat dle UX návrhu, což si myslím, že se mi povedlo a s celkovým vzhledem byl spokojen i můj konzultant.

Při tvorbě komponent jsem také narazil na několik problémů, hlavním problémem byl ESLint, který testuje kód dle předdefinovaných pravidel, několikrát se mi stalo, že určitá část kódu neprošla testem a následné opravy nebyly pro mne jako úplného neznalce React a JS jednoduché. Dalším problémem byla práce s GITem, jelikož do kódu zasahovalo více lidí naráz, neobešel jsem se bez rebase (integrace změn z jedné větve do druhé - přeskládání) nad master branch (hlavní větev), což byla někdy dost komplikovaná záležitost. Ale i přes všechny ty problémy se vše podařilo.



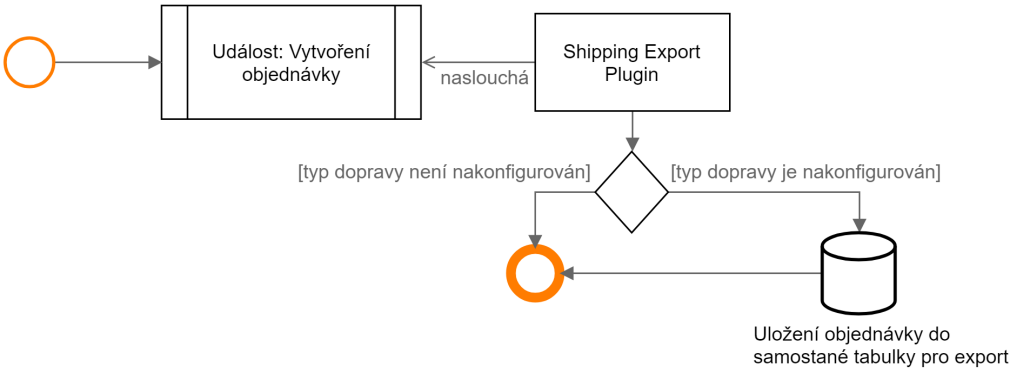
Obrázek 12: Výsledný vzhled přehledu zásilek

6 Sylius modul

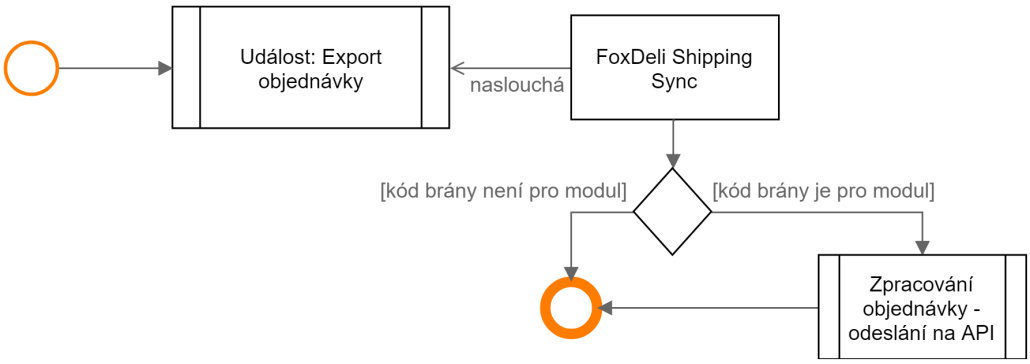
Před samotnou tvorbou modulu a seznámení s FoxDeli API bylo potřeba provést analýzu postupu a výběru vhodných prostředků. Při analýze možností Sylius frameworku jsem narazil na problém, který by mi znemožnil integraci FoxDeli API. Jednalo se hlavně o problém s rozšiřitelnosti Sylius o vlastní dopravu. Aktuálně podporuje pouze přidání nového způsobu přepravy a jeho nastavení jako je cena, kategorie přepravy, výpočet přepravním poplatků, dostupnost dopravy pro určitou zemi apod. Neobsahuje žádnou logiku pro práci s externími API nebo exportem zásilek.

Při hledání řešení mého problému jsem nalezl modul Shipping Export Plugin (v oficiálním repositáři ověřených modulů), který do Sylius přidá abstraktní vrstvu, která umožňuje integrovat vlastním API dopravy, včetně konfigurace a exportu zásilek. Po analýze kódu modulu a následné instalaci jsem došel k závěru, že to bude ideální řešení pro mou integraci.

Modul Shipping Export Plugin funguje na principu naslouchání událostí. Poté co je objednávka uživatelem potvrzena, dojde k vyvolání Sylius události "nová objednávka" a tím k zavolání funkce modulu, ve které dojde k ověření, zda daná objednávka obsahuje způsob dopravy, který má modul ve své konfiguraci. V případě, že ano dojde k uložení čísla objednávky do samostatné tabulky v databázi, v opačném případě se objednávka ignoruje. Tyto objednávky je poté možné vyexportovat. Při exportu dojde k vyvolání události, na kterou naslouchají moduly (vlastní integrace API), které poté zpracují danou objednávku - odešlou data na externí API, získají štítek nebo číslo zásilky. Po zpracování objednávky dojde k uloží získaného štítku / čísla zásilky k dané objednávce.



Obrázek 13: Proces zpracování/příprava objednávky Shipping Export Plugin modulem



Obrázek 14: Proces zpracování objednávky modulem pro integraci FoxDeli API

6.1 FoxDeli API

FoxDeli API je postaveno na RESTové architektuře tzn. datově orientované API, které určuje jak přistupovat k datům. Pro autorizaci se používá protokol OAuth 2.0, který funguje na principu registraci aplikace a výměny tokenů (klíčů).

Rozhraní pro práci se zásilkami je velice jednoduché. Nejdříve je potřeba získat přístup k API, k tomu slouží *client_id, client_secret a refresh_token*, po úspěšně provedení autorizačního požadavku získáme *access_token*, který přidáme do hlavičky každého požadavku a tím nám API umožní pracovat se zdrojem, ke kterému máme přístup. API obsahuje několik zdrojů a metod, pro mé potřeby jsem použil zdroj deliveries a metody POST, PATCH a GET - pro vytvoření zásilky, uzavření zásilky a získání štítku. [27]

Obrázek 15: Schéma procesu FoxDeli API - zdroj deliveries

Metody zdroje deliveries:

- POST - /deliveries - import nových zásilek
- PATCH - /deliveries - uzavření zásilek
- GET - /deliveries/tickets?deliveryid=XXX&position=1&printFormat=single
  - deliveryid - číslo zásilky
  - position - pozice štítku
  - printFormat - default (A4), single (koutouč)

6.2 Struktura modulu

Při tvorbě modulu jsem musel dodržovat určitá pravidla Symfony frameworku. Jednalo se hlavně o názvy namespace a adresářovou strukturu.

Struktura:

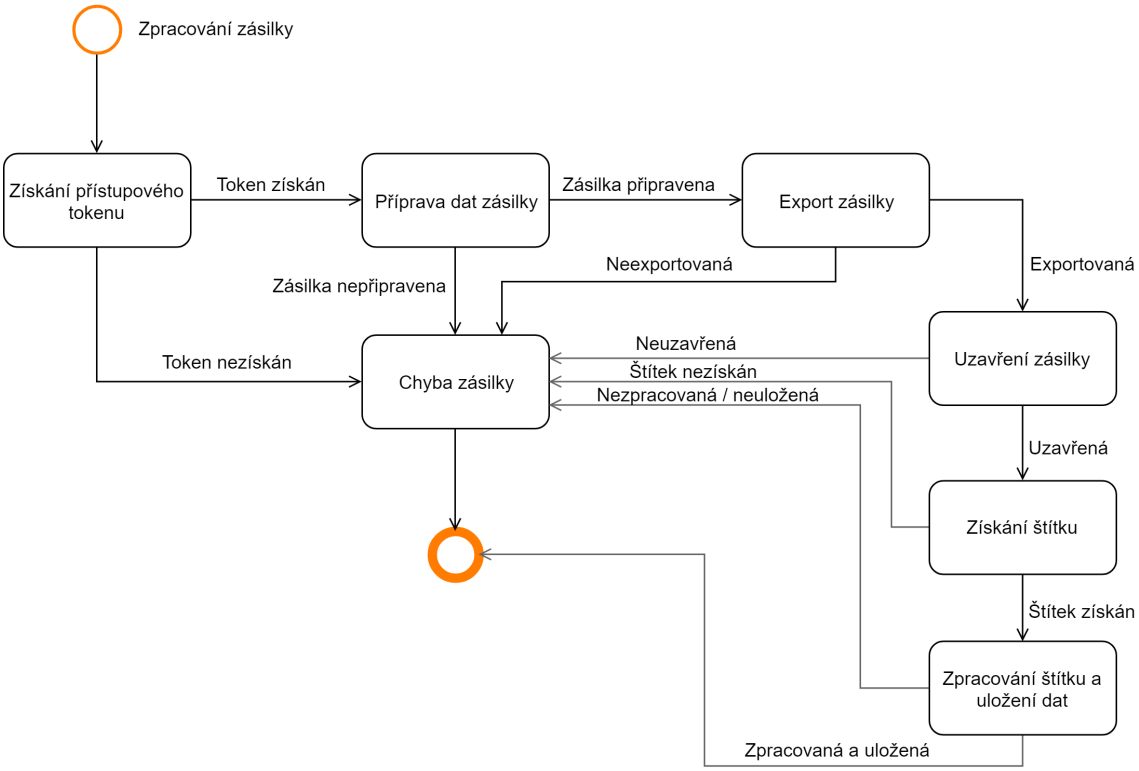
- src
  - DependencyInjection
    - FoxDeliShippingSyncExtension.php - načtení konfiguračních souborů
  - Events
    - EventListener.php - naslouchání na událost Shipping Export Plugin
  - Forms
    - ShippingGateway.php - přidání konfiguračního formuláře do Shipping Export Plugin
  - Resources
    - config
      - services.yml - registrace služeb a událostí
    - translations - překlady
      - messages.cs.yml
      - messages.cs_CZ.yml
      - messages.en.yml
      - messages.en_GB.yml
      - messages.en_US.yml
  - Services
    - FoxDeliService.php - FoxDeli API klient
  - FoxDeliShippingSync.php - hl. soubor modulu
- composer.json - soubor balíčku (název, verze, autoload, ...)

6.3 Implementace

Samotná implementace FoxDeli API probíhala v rámci třídy *FoxDeliService*. Komunikaci s API jsem rozdělil do několika funkcí - *getAccessToken, sendDelivery, completeDelivery a getDeliveryLabel*. Dále jsem si vytvořil několik pomocných funkcí například pro naformátování dat zásilky dle požadavků API, anebo pro inicializaci HTTP klienta a nastavení globální hlavičky.

Proces zpracování zásilky probíhá v několika krocích:

1. Získání přístupového tokenu - *getAccessToken*
2. Příprava zásilky k exportu (naformátování dat)
3. Export zásilky - *sendDelivery*
4. Uzavření zásilky - *completeDelivery*
5. Získání štítku - *getDeliveryLabel*
6. Zpracování štítku a uložení dat



Obrázek 16: Proces zpracování zásilky

Popis funkcí:

- getAccessToken - Slouží k získání přístupového tokenu (*access_token*), který je poté nutné vložit do hlavičky každého požadavku, bez tohoto tokenu není možné komunikovat s API.
- sendDelivery - Vstupní funkce pro odeslání dat zásilky. Po úspěšně vykonaném požadavku dojde k vytvoření zásilky ve FoxDeli. Odpověď obsahuje veškeré informace o nově vložené zásilce, důležité je číslo zásilky, které je potřeba pro další zpracování.

```
$guzzleParams = [
    ...
    'form_params' => $this->prepareDeliveryData(),
];
$response = $this->guzzleClient->post("{$this->config['endpoints']['
    api']}/deliveries", $guzzleParams);
```

Výpis 2: Ukázka požadavku pro import zásilky

- completeDelivery - Aby bylo možné získat štítek k dané zásilce, je nutné ji nejprve uzavřít. Tato funkce zašle požadavek na uzavření zásilky s číslem zásilky, které bylo získáno v předchozí funkci. Odpověď pouze obsahuje, zda se požadavek vykonal/nevykonal.

```
$guzzleParams = [
    ...
    'form_params' => [ 'deliveries' => [ [ 'deliveryId' => $deliveryId,
        'closed' => true ] ] ]
];
$response = $this->guzzleClient->patch("{$this->config['endpoints']['
    api']}/deliveries", $guzzleParams);
```

Výpis 3: Ukázka požadavku pro uzavření zásilky

- getDeliveryLabel - Po uzavření zásilky je možné získat její štítek. Funkce zažádá o štítek k zásilce zpracované v předchozích funkcích. Výsledkem je odpověď obsahující štítek v Base64 kódování - po dekódování přes funkci *base64_decode* získáme PDF soubor.

```
$response = $this->guzzleClient->get("{$this->config['endpoints']['api
    ']}/deliveries/tickets?deliveryId={$deliveryId}&position=1&
    printFormat=single", $guzzleParams);
```

Výpis 4: Ukázka požadavku pro získání štítku zásilky

6.4 Tvorba balíčku

Aby bylo možné modul použít v Symfony frameworku je nutné vytvořit tzv. *composer.json* soubor, ve kterém jsou obsaženy základní informace o balíčku - název, typ balíčku (sylius-plugin), popis, a také požadavky balíčku a závislosti - nejnižší podporovaná verze PHP, závilost na Sylius framework a v mém případě také závislost na Shipping Export Plugin. Další důležitou součástí je tzv. *autoload* neboli automatické načtení tříd balíčku, které budou poté k dispozici v rámci celé aplikace.

Poté je možné balíček použít v Symfony frameworku. Do *composer.json* souboru stačí připsat závislost na mém modulu, tím dojde ke zpřístupnění balíčku v rámci aplikace.

```
{
  "name": "itlab/sylius-foxdeli-shipping-sync",
  "type": "sylius-plugin",
  "description": "FoxDeli shipping sync plugin for Sylius.",
  "require": {
    "php": "^7.2",
    "sylius/sylius": "^1.0",
    "bitbag/shipping-export-plugin": "^1.2.6"
  },
  "autoload": {
    "psr-4": {
      "ITLab\\FoxDeliShippingSyncPlugin\\": "src/"
    }
  },
  ...
}
```

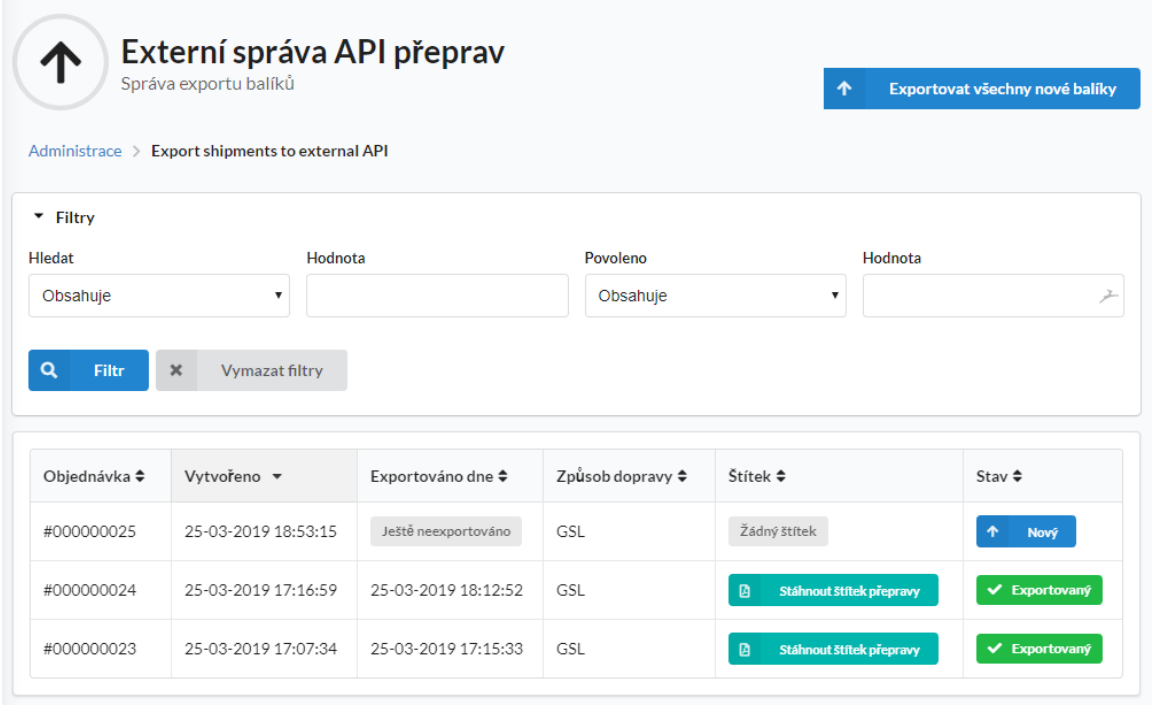
Výpis 5: Ukázka informací o balíčku

6.5 Shrnutí

Z počátku jsem měl problém s přístupem k FoxDeli API, z důvodu neznalosti OAuth2 protokolu jsem nevěděl jak získat *refresh_token*, který byl potřeba k získání dočasného *access_tokenu*, abych mohl mohl s API komunikovat. Nicméně po zjištění detailnějších informací o OAuth2 jsem úspěšně získal potřebný *refresh_token* a mohl tak začít s testováním a následnou implementací FoxDeli API klienta. Při implementaci modulu jsem bohužel narazil na další problém a to na správné pojmenování souborů a *namespace*. Naštěstí po konzultaci s kolegou se vše vyřešilo.

Cíl propojit Sylius s FoxDeli API se podle mě podařil, a i přes problémy, které jsem v průběhu měl, si myslím, že se mi podařilo efektivně, a dá se říct snadno propojit pro mě neznámou e-shop platformu s neznámým API.

Samozřejmě by se dal modul i více vylepšit například přidáním tlačítka pro export přímo v detailu objednávky nebo mít možnost vyexportovat objednávky pouze vybraného způsobu přepravy. Bohužel pro tato vylepšení již nezbyl čas.



Obrázek 17: Sylius - export zásilek (Shipping Export Plugin)

7 Závěr

7.1 Uplatnění teoretických a praktických znalostí a dovedností získaných během studia

Za dobu studia jsem měl možnost získat spoustu znalostí a dovedností, bez kterých bych se při vykonávání odborné praxe neobešel. Nejvíce užitečným předmětem byl pro mne Vývoj internetových aplikací, ve kterém se vyučovaly technologie HTML, CSS a JavaScript. Znalost těchto technologií mi byla užitečná ve větší části praxe. Další užitečným předmětem byl Vývoj informačních systému, ve kterém jsem získal hlubší znalosti o principech při tvorbě aplikací.

Základní princip programování je ve všech jazycích skoro stejný, liší se pouze v maličkostech a syntaxí, která bývá často podobná, z toho důvodu bych zde chtěl zmínit přínos předmětů Programování I a Programování II, které mi poskytly všeobecné základy principů a paradigmat programování v oblasti objektově orientovaného programování.

7.2 Znalosti a dovednosti scházející v průběhu vykonání odborné praxe

Chybějící znalosti a dovednosti, které by mi pomohly a ulehčily tak průběh mé praxe je mnoho. Hlavním důvodem jejich absence byla neznalost pokročilých technologií pro vývoj webových aplikací. I přes to, že se částečně věnuji vývoji webových aplikací, většinou jsem se těmto technologiím vyhýbal z důvodu časové náročnosti se je naučit a mezi mé hlavní zájmy patří spíše back-end vývoj.

V průběhu praxe mi nejvíce chyběly zkušenosti s JavaSriptovým framworkem React a také hlubší znalosti SASS. Také by mi pomohlo, kdybych měl větší přehled o PHP frameworku Symfony, který používá jako základ e-commerce framework Sylius. S těmito technologiemi jsem se setkal až v průběhu praxe, z toho důvodu nebyla práce s nimi příliš jednoduchá.

7.3 Přínos a celkové zhodnocení odborné praxe

Během praxe jsem se naučil spoustu nových technologií, zejména JavaScriptový framework React, který je často používaný velkými projekty a funguje na podobném principu jako frameworky Vue.js nebo Angular, které jsou také velmi rozšířené.

Vyzkoušel jsem si také práci back-end vývoje v jazyce PHP při integraci FoxDeli API do e-commerce frameworku Sylius, čímž jsem si rozšířil znalosti a získal nové zkušenosti s dalšími platformami postavené na jazyce PHP.

Celkově absolvování praxe ve firmě IT Lab czech s.r.o. hodnotím kladně, měl jsem možnost pracovat ve skvělém kolektivu, poznat spoustu nových nástrojů pro tvorbu a správu větších projektů, a také jsem se mohl zdokonalit ve front-end i back-end programování.

Literatura

[1] Wikipedia. *List of server-side JavaScript implementations*. 2019. URL: https://en.wikipedia.org/wiki/List_of_server-side_JavaScript_implementations.

[2] Wikipedia. *JavaScript*. 2019. URL: https://cs.wikipedia.org/wiki/JavaScript.

[3] Wikipedia. *HTML*. 2019. URL: https://en.wikipedia.org/wiki/HTML.

[4] Computer Hope. *What is HTML (Hypertext Markup Language)?* 2018. URL: https://www.computerhope.com/jargon/h/html.htm.

[5] Wikipedia. *Cascading Style Sheets*. 2019. URL: https://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[6] Mozilla. *Introduction to CSS*. 2019. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS.

[7] Wikipedia. *Sass (stylesheet language)*. 2019. URL: https://en.wikipedia.org/wiki/Sass_(stylesheet_language).

[8] Wikipedia. *Bootstrap (front-end framework)*. 2019. URL: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework).

[9] JS Foundation. *About - ESLint*. 2019. URL: https://eslint.org/docs/about/.

[10] FlavioCopes. *An introduction to Yarn*. 2019. URL: https://flaviocopes.com/yarn/.

[11] Facebook Inc. *React - A JavaScript library*. 2019. URL: https://reactjs.org/.

[12] Wikipedia. *React (JavaScript library)*. 2019. URL: https://en.wikipedia.org/wiki/React_(JavaScript_library).

[13] DžejEs. *React - Úvod*. 2019. URL: https://www.dzejes.cz/react-uvod.html.

[14] Dan Abramov a the Redux documentation authors. *Redux - A predictable State Container for JS Apps*. 2019. URL: https://redux.js.org/.

[15] Wikipedia. *Redux (JavaScript library)*. 2019. URL: https://en.wikipedia.org/wiki/Redux_(JavaScript_library).

[16] Meteor Development Group Inc. *Introduction*. 2019. URL: https://www.apollographql.com/docs/react/.

[17] Facebook Inc. *GraphQL - A query language for your API*. 2019. URL: https://graphql.org/.

[18] GraphQL. *GraphQL je dotazovací jazyk pro vaše moderní API*. 2019. URL: http://graphql.cz/graphql-dotazovaci-jazyk/.

[19] BabelJS. *Babel - The compiler for next generation JavaScript*. 2019. URL: https://babeljs.io/.

[20] Webpack. *Webpack*. 2019. URL: https://webpack.js.org/.

[21] Wikipedia. *Webpack*. 2019. URL: https://en.wikipedia.org/wiki/Webpack.

[22] Git. *Git*. 2019. URL: https://git-scm.com/.

[23] Wikipedia. *Git*. 2019. URL: https://en.wikipedia.org/wiki/Git.

[24] Symfony SAS. *What is Symfony*. 2019. URL: https://symfony.com/what-is-symfony.

[25] Wikipedia. *Symfony*. 2019. URL: https://en.wikipedia.org/wiki/Symfony.

[26] Sylius. *Sylius eCommerce Framework*. 2019. URL: https://sylius.com/sylius-ecommerce-framework/.

[27] Balíkonoš.cz. *Balíkonoš API - dokumentace RESTful zdrojů*. 2019. URL: https://test.balikonos.cz/doc-api/resources/.